开源项目中的补丁添加过程研究

摘要：补丁添加过程对开源系统的质量有着重要影响。本文提出了一个组织良好的开源项目中补丁添加过程。通过对现有文献以及开源项目的观察和研究，总结出了补丁添加过程中涉及的 5 个环节。

关键词：开源项目　补丁　质量

分类号：C931.2

A Study on the Process of Patch-Adding in Open Source Projects

Abstract： Patch-adding process affects the quality of OSS systems. The paper proposes a well-organized patch-adding process in OSS project. By observation and by examination of the current literature and some OSS projects, we conclude five steps in the process.

Keywords:　Open Source Projects　Patch　Quality

1. 引言

开源软件项目的数量和开源系统的规模近年来都呈指数增长。开源项目的成功在很大程度上依赖于开源系统的质量和开源社区。而核心的开发人员则被认为是社区里的精英，因为他们的贡献直接影响开源系统的质量和发展。但是任何一名精英无论其背景如何，都是从对开源系统提交补丁开始的。所谓补丁（patch）是对开源项目源代码或文档的修改，它在被放进最终的软件版本之前要经历几个阶段。这些阶段就构成了补丁添加的过程。补丁添加过程对开源软件系统的质量同样有着重要影响。因为，首先，它是开源系统主要的质量保证机制。一个好的补丁添加过程包括补丁审核活动，防止不合格的补丁进入源代码，其次，补丁提交为社区招募潜在开发者提供了机会。通过补丁提交，项目管理者可以看到提交者的能力和对社区的奉献精神。虽然补丁提交很重要，但是对其的研究并不多，大部分的研究工作都集中在补丁审核这一部分。

本文将对一些成功开源项目网站和社区进行调研，分析和比较各项目补丁提交过程，从而识别过程中的关键活动。一般来说，补丁提交的过程都并没有清晰地记录在项目文档里，因此需要对邮件列表和论坛里的活动信息进行分析，总结出补丁添加的过程，以及一些优秀实践，针对当前补丁添加过程中存在的普遍问题，结合自己对 OSS 项目补丁添加过程的看法和建议，总结出一个管理组织良好的补丁添加过程的架构。

2. 补丁添加过程存在的问题

所谓补丁是指对 OSS 项目现存代码库或文档的修改，而大部分 OSS 项目都是以补丁形式接收代码。补丁添加过程中的 patch 审核是 OSS 开发中最重要的活动之一，因为它确保补丁满足项目标准。开源开发的异步性也令这种质量保证机制变得很实用，因此补丁添加过程极大地影响 OSS 开发质量。然而，目前在补丁添加过程中普遍存在 3 个主要的困难：

（1） 补丁的审核过程耗时而且速度很慢。于是有一些项目如 Apache web 服务器项目就做了一些改善措施，Apache 要求每个补丁在被采用到项目代码库前需要被审核，但是允许 committers（具备提交权限的开发

者）在审核前直接将补丁提交到代码库。但是这样做却会为 OSS 质量带来风险。

（2） 不同的开源项目补丁审核也不一样，当开发者同时参与了多个项目时，审核过程和工具的不同可能会造成混乱，会需要更多的时间去学习。Asundi 对 5 个开源项目进行了研究，并描述了补丁提交-审核过程，但是其研究结果还不足以了解和识别补丁添加过程的关键点。

（3） 补丁经常丢失和被遗忘。在 OSS 项目中没有被审核的补丁数量大概占提交补丁总量的 27%~54%。[1]在 Apache 项目中，同行评议过程是强制性的，但仍然有 23%的补丁被提交者（committers）忽略，8%的已被提交的补丁未被审核。[2]

（4） 在补丁添加过程过程中大多数都被拒绝掉，而被拒的常见原因涉及补丁设计和格式问题[3]。而且在开源社区中，补丁开发者的数量显著超过审核者的数量，[4]那么如何提高补丁被审核者发现、采纳的几率也是有关补丁设计和格式等的问题。

3. 补丁添加过程

虽然不同的 OSS 项目它具体的补丁添加过程也不同，但是仍然存在一些共同的地方。在概念层级上，不同 OSS 项目的补丁添加过程是相似的，补丁提交的工具和方式，人员角色以及过程中的活动均是相似的。

3.1 补丁提交的工具和方式

提交补丁是为了修复错误，维护代码，创建新功能，增强原有功能等。补丁有三种提交的方式：电子邮件、问题跟踪系统中问题报告的一部分以及补丁跟踪系统作为一个补丁提交。

问题报告包含与软件问题相关的信息，可以是软件的错误，新功能等。问题报告一般存放在问题跟踪系统或者错误跟踪系统中，问题跟踪系统允许与问题解决相关的补丁附加在问题报告中，并且支持对补丁或者问题报告本身进行评论。在问题跟踪系统中的问题报告数量很多，但是仅仅只有一小部分问题报告包含补丁。

补丁跟踪系统可以取代问题跟踪系统来提交、管理和跟踪 patch。它的特点是可以根据 patch 的状态来组织 patch，对 patch 发表评论等。

许多 OSS 项目因为不同的目的有多个邮件列表。最常见的邮件列表有支持服务邮件列表、开发者邮件列表、软件问题邮件列表以及提交邮件列表等。软件问

---

[1] Asundi,J.and Jayant,R.,"Patch Review Processes in Open Source Software Development Communities: A Comparative Case Study,"[C]//Proceedings of the 40th Annual Hawaii International Conference on System Sciences,IEEE Computer Society,p.166c,2007

[2] Rigby,P.C.,German, D.M. and Storey,.M., "Open Source Software Peer Review Practices: A Case Study of the Apache Server,"[c]//Proceedings of the 30th international conference on Software Engineering, Leipzig,Germany:ACM,2008:541-550

[3] Nurolahzade, M., Nasehi, S.M., Khandkar, S.H. and Rawal, S., "The role of patch review in software evolution: an analysis of the mozilla firefox," Proceedings of the joint international and annual ERCIM workshops on Principles of software evolution (IWPSE) and software evolution (Evol) workshops, Amsterdam, The Netherlands: ACM, pp. 9-18, 2009.

[4] Weibgerber,P., Neu, D. and Diehl, S., "Small Patches get in !,"[C]// Proceedings of the 2008 international working conference on Mining software repositories, Leipzig, Germany: ACM, pp,67-76,2008

题邮件列表用来记录项目问题跟踪系统中发生的变化。提交邮件列表可以记录源代码库中发生的变化。软件问题邮件列表和提交邮件列表都由相应的系统自动更新，并且只能读。开发者们可以在开发者邮件列表里讨论有关源代码发生的变化，并且许多 OSS 项目也都是通过开发者邮件列表来执行修复程序审核的。Patch 以邮件的方式进行提交，社区通过回复邮件对其进行评论。

3.2 补丁提交过程中的角色

在补丁提交过程中涉及到多种角色，比如问题报告者、补丁开发者、其他开发者、补丁提交者以及补丁审核者。问题报告者通过开发者邮件列表或者问题跟踪系统向社区报告软件存在的问题。补丁开发者通常没有权限直接修改项目的源代码库，补丁提交者才有这个权限。社区里的其他开发者们可以对提交的补丁进行反馈。补丁在添加到源代码库前，必须经过补丁审核者的检查。

3.3 补丁提交过程中的各环节

通过调研，我们总结出了 5 个环节，这些环节包括创建补丁、公布补丁、发现补丁、审核补丁以及应用补丁。

（1）创建补丁

创建补丁在 OSS 项目中有两种方法，对于简单的补丁，补丁开发者可以使用 diff 工具来创建补丁，对于将会修改多个文件的补丁，补丁开发者可以通过版本控制客户端比如 CVS、SVN、Git 等提供的命令创建。

一些开源软件项目在此阶段的一些质量保证措施：

a. 一般来说开源软件项目都会要求补丁必须遵守该项目的编码标准。所谓编码标准是描述 OSS 项目首选编码风格的文档。
b. 在补丁格式方面，基本都大同小异。由于补丁没有足够的背景信息很难理解，Mozilla 项目使用 8 行文本对其进行描述，这样补丁审核者不用打开源文件就能理解补丁用途。
c. 在 Linux 项目中要求每个补丁只表达一个逻辑改变，而不是同一个补丁中包含若干个变化，这样使得补丁间变得更独立，而且可能更小，从而方便审核者进行审核。

结合开源软件项目成功的经验，在这个过程中补丁开发者应该编写、测试被修改的代码，检查是否与项目编码标准相符，以及根据 OSS 项目采用的许可证检查是否有权共享代码。当创建补丁时，补丁开发者需要遵循 OSS 项目的规则，并且为了便于审核，一个补丁最好只含有一个逻辑变化。补丁格式要符合 OSS 项目标准，这对于增加补丁被采用的几率非常重要。补丁创建完，补丁开发者还需要将补丁应用到最新的代码版本上进行测试。由于补丁常常被修改，因此为了避免混乱和丢失补丁，补丁应该唯一命名并且包含版本号作为补丁名字的一部分。

为了增加补丁被接受的可能性，一般说来，补丁至少需要一段关于它的用途以及为什么它应该被添加到源代码中的描述说明。如果该补丁是对错误跟踪系统中的一个错误的修复，那么该补丁应该还应该包含被修复错误的标识号。例如 PhpMyAdmin 要求补丁里包含项目的名字，补丁的状态（修复中，已完成），测

试结果，怎样使用此新功能以及该变化如何影响性能的描述。因为如果补丁已经包含有足够多的相关信息，审核者在审核时就不用耗费精力去了解关于该补丁的信息，这将大大增加其被审核的几率。一个好的包含足够多信息的补丁也会使得后面的审核过程更加有效率。

（2）公布补丁

在公布补丁前，首先要选择一个公布渠道或方式。主要存在三种公布方式：项目邮件列表、问题/错误跟踪系统以及补丁跟踪系统。有的项目提供一种渠道，有的提供多种渠道，比如 Linux Kernel 项目仅仅使用邮件列表作为唯一的提交渠道。FreeMind 同时提供邮件列表和补丁跟踪系统两种渠道。

选定补丁公布渠道后，补丁开发者需要根据项目要求进行打包，通过选定的渠道提交出去，等待社区的反馈。公布步骤随提交工具的不同而不同。

问题跟踪系统是最常见的提交渠道，OpenOffice.org、Eclipse 以及 Mozilla 均是使用 Bugzilla。使用这种工具时，用户可以在现有的问题报告上附加一个补丁或者重新创建一个新的问题报告，如果已经存在一个有关的问题报告时，补丁开发者应该将该补丁附加到问题报告上，否则重新创建问题报告。

电子邮件是最古老的提交渠道。许多 OSS 项目已经不再使用这种提交工具。使用这种方式提交补丁也比较简单。Linux 项目要求将 patch 内容放在邮件的正文中。而 PostgreSQL 则规定将 patch 文件作为邮件附件提交。Patch 开发者在将 patch 提交到软件问题邮件列表的同时，也需要将其提交到开发者邮件列表供其他开发者评论。

补丁跟踪系统是一个基于 web 的应用，旨在支持 patch 的审核过程。补丁跟踪系统方便了补丁开发者和审核者提交、跟踪、搜索和审核。Android 项目使用了补丁跟踪系统辅助补丁审查和沟通。Linux 项目使用它记录 patch 的状态。PostgreSQL 将补丁跟踪系统与电子邮件归档库集成，用户可以利用它检查 patch 状态以及给予反馈。

（3）发现补丁

这个过程将会涉及补丁开发者、审核者、其他开发者以及补丁提交者多个角色。通过对这个环节的改进和良好管理可以改善许多补丁丢失或者被忽略遗忘的情况。具体就需要补丁开发者、审核者及其他开发者们能够随时跟踪到补丁，并且能有一个良好的方式及时让审核者知晓新的补丁已经被补丁开发者提交。通过文献调研我们发现存在着多种审核者发现补丁的方式，补丁开发者选择的公布渠道有着重要影响。

如果补丁是通过邮件列表直接发送给审核者，审核者通过接收邮件就可以发现补丁。但是补丁开发者却没有办法知道审核者是否已经发现补丁，除非审核者回复该邮件，但是审核者回复邮件确认已收到的情况不太可能发生。于是补丁开发者就只能等直到审核者有时间阅读并且给予反馈，这个过程可能需要很长一段时间。同时，如果使用邮件列表作为唯一的提交方式的话，就需要开发社区的审核者管理那些包含补丁的邮件。结果由于审核者需要花大量的时间管理和发现补

丁，审核者可能只会从众多补丁中挑选一些进行审核，导致其他补丁被忽略和遗忘。

如果被提交到问题跟踪系统或者补丁跟踪系统，审核者可以通过查看问题报告发现。如果这些系统有发送邮件的功能的话，当报告被修改了的时候，审核者将会得到通知。于是审核者可以通过查看邮件发现补丁，但是这种情况同样会发生上述适用邮件列表提交的问题。

但如果 OSS 项目能提供一种方式可以集中显示所有活跃的补丁列表，比如 Drupal 项目就在它的主页上提供了到活跃 patch 列表的链接。审核者就很容易发现新提交的补丁，补丁开发者也能在该列表上追踪补丁的状态。同时如果 OSS 项目能为审核规定一个时间段，这样一旦过了规定的时间段，补丁开发者仍然没有收到反馈，可以再次提交。

（4）审核补丁

这个过程同样会涉及到补丁开发者、审核者、其他开发者以及补丁提交者。因为审核补丁的目的在于决定是否将补丁采用到项目的源代码中，这对于最后软件产品的质量将产生直接的影响。因此这个过程显得尤为重要。这个过程主要涉及三个环节：检验补丁，完善补丁以及解决补丁。

A 检验补丁

社区内任何开发者都可以检验补丁的质量并且提出一些提高质量的建议。审核的重点在于确保补丁有资格采用到项目的代码库中。审核者基于补丁的质量、安全性、可维护性等技术指标对其审核，检验其是否遵循项目的标准，补丁引入的缺陷是否会在将来导致整个软件系统产生问题，该补丁是否是比较好的解决问题的方法，补丁是否遵循代码授权许可等等。

在 OSS 项目中用到的最常见的检验技术是代码审核。代码审核一般由有经验的开发者来执行，这些开发者一般是技术专家或者是了解该方面知识的领域专家。审核者在验证完补丁的质量后提出改进的建议。

但是由于 OSS 社区每天接收补丁的数量很多，因此需要审核者尽可能快地对补丁进行审核，或许有经验的审核者通过简单的审查就可以发现存在的问题，但是对于大多数审核者来说并非那么容易。如果借助代码审核工具进行审核将会大大方便审核者们的工作，提高工作效率。目前有许多基于 web 的代码审查工具，他们可以追踪待审代码的改动，所有更改的代码都会被高亮，并且可以写评论，工具将会记录这些评论等功能 。但是根据调研发现，利用此类工具的 OSS 项目不多，发现的仅有 Android 使用代码审查工具支持代码审查活动。另外具体的对补丁测试的技术将在后面详细论述。

B．完善补丁

补丁开发者可以根据审核者提出的意见对补丁进行完善改进。补丁完善的过程是一个迭代循环的过程。补丁开发者也许要在补丁被采纳之前修改许多次。因此为了使得审核者了解修改的历史，建议补丁开发者最好在再次提交补丁新版本时，对代码发生的变化写一个总结。提高审核者对补丁的了解程度，那么最终被

审核者采用或者改进的几率就会越大。

（5）应用 patch

对补丁最后的处置解决方式有两种：拒绝或者接受。不同的 OSS 项目有不同的补丁解决方式。大多数情况下，对补丁的决议结果都是由负责源代码相关部分的项目核心开发者作出的。Linux 使用分阶段逐步审批的方法，一个补丁要顺着项目层级结构从子系统负责人开始经过多次审核。Apache 则采用投票方式决议是否接受补丁，只有有 3 个赞成票，没有反对票的情况下补丁才能被采纳。Mozilla 项目则是有两个审核者进行审核，一个审核者是受该补丁影响的模块的负责人，另一审核者则是更高级别的审核者，它负责研究补丁可能会产生的更高层级的影响。

负责受补丁影响模块的任何一个提交者都可以将经过审核的补丁应用到项目代码库中。大多数情况下参与代码审核的人中会有一个补丁提交者，他可以将补丁提交到代码库中，审核者也可以进行提交。有可以自动对补丁进行提交的工具，但是大部分 OSS 项目仍然半自动的方式进行提交。

这个过程存在着已经经过审核的补丁，却没有人进行提交的情况。对这种情况，Mozilla 项目对每一个通过审核的补丁添加一个特定的关键字，从而使得这些补丁可以很容易地被提交者们从错误跟踪系统中检索出来，进行提交。PostgreSQL 的 CommitFest 提供一个准备被提交的补丁列表。提交者们浏览列表然后提交。

在对 patch 进行提交时，提交者至少需要一个补丁文件和版本管理系统（如 CVS）。首先，提交者将 patch 应用到本地的源代码中，再使用版本管理系统将其提交到源代码库中。Patch 应用到源代码库这个过程是自动进行的，比如 Android 项目中，就是由补丁跟踪系统自动应用补丁。

4. 总结

开源软件项目可以说是通过不断添加补丁的过程进行开发的，那么在这个过程中的低效率以及失误、故障也将会对其开发质量产生消极影响，因此 patch 添加过程的良好设计和优良管理是影响开源软件项目的开发质量的关键因素。虽然不同的 OSS 项目采用不同的补丁添加过程，但是他们之间仍然存在一些共同点。然而以往对于补丁添加的研究主要集中于一两个环节，对于整个补丁添加生命周期的研究将使大量的开源项目受益。

参考文献

[1] Asundi,J.and Jayant,R.,"Patch Review Processes in Open Source Software Development Communities: A Comparative Case Study,"[C]//Proceedings of the 40th Annual Hawaii International Conference on System Sciences,IEEE Computer Society,p.166c,2007

[2] Rigby,P.C.,German, D.M. and Storey,.M., "Open Source Software Peer Review Practices: A Case Study of the Apache Server,"[c]//Proceedings of the 30th international conference on Software Engineering, Leipzig,Germany:ACM,2008:541-550

[3] Nurolahzade, M., Nasehi, S.M., Khandkar, S.H. and Rawal, S., “The role of patch review in software evolution: an analysis of the mozilla firefox,” Proceedings of the joint international and annual ERCIM workshops on Principles of software evolution (IWPSE) and software evolution (Evol) workshops, Amsterdam, The Netherlands: ACM, pp. 9-18, 2009.

[4] Weibgerber,P., Neu, D. and Diehl, S., “Small Patches get in !,”[C]// Proceedings of t he 2008 international working conference on Mining software repositories, Leipzig, Germany: ACM, pp,67-76,2008

[5] Tobias Otte, Robert Moreton, Heinz D.Knoell. Applied Quality Assurance Methods under the Open Source Development Model. IEEE International Computer Software and Applications Conference,2008

[6] Dindin Wahyudin, Alexander Schatten, Dietmar Winkler, Stefan Biffl. Aspects of Software Quality Assurance in Open Source Software Projects:Two Case Studies from Apache Project[C]//33rd EUROMICRO Conference on Software Engineering and Advanced Applications,2007:229-236

[7] Rigby PC, German DM, Storey M A.Open Source Software PeerReview Practices: A Case study of the Apache Server[EB/OL][2011-06-19]. http://helium.es.uvic.ca/pr/submit_rigby_icse.pdf

[8] Wahyudin.D, Schatten.A,Winkler,D et al. Aspects of Software Quality Assurance in Open Source Software Projects: Two Case Studies from Apache Project[C]//33rd Euromicro Conference on Software Engineering and Advanced Applications,2007(19):229-236

[9] Wahyudin.D, Schatten.A,Winkler,D et al. Aspects of Software Quality Assurance in Open Source Software Projects: Two Case Studies from Apache Project[C]//33rd Euromicro Conference on Software Engineering and Advanced Applications,2007(19):229-236

[10] Tawileh,A. Rana,O. Free and Open Source Software Quality Assurance[C]//IEEE International Conference on Industrial Engineering and Engieering Management,2009:1386-1390

[11] Conley, C.A., Sproull, L. Easier said than Done: An Empirical Investigation of Software Design and Quality in Open Source Software Development.42nd Hawaii International Conference on System Sciences,2009:1-10

[12] Asundi, J. and Jayant, R., “Patch Review Processes in Open Source Software Development Communities: A Comparative Case Study,” Proceedings of the 40th Annual Hawaii International Conference on System Sciences, IEEE Computer Society, p. 166c, 2007.